REMARKS

Claims 1-30 are pending in the present application.

Claim 15 stands rejected under 35 U.S.C §101 as being directed to non-statutory subject matter. As shown above, Applicant has amended claim 15 to overcome the rejection.

The Examiner objected to the drawings and asserted the Applicant's specification includes a reference designator that is not in the drawings. Specifically, the Examiner indicated the specification at page 2 paragraph [0023] includes a reference to "a service processor 113," which is not included in the drawings. Applicant has searched the entire specification, including paragraph [0023], and cannot find any reference number 113. Applicant notes the reference designator 113 is used in the Crowell are cited by the Examiner. Applicant did however find an erroneous reference to service processor 100B on page 15 in paragraph [0042]. As shown above, Applicant has amended the specification to correct the error. All other service processor references appear to include reference number 150 or 250, which are both in the drawings. Applicant respectfully requests the Examiner notify Applicant if this is not the error in question.

Claims 1-5, 7, 9, 10, 12, 15-21, 23, 25, 26, and 28 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Crowell (U.S. Patent Publication No. 2004/0215440) (hereinafter "Crowell") in view of Huang et al. (U.S. Patent No. 5,267,246) (hereinafter "Huang"). Applicant respectfully traverses at least portions of this rejection.

Claims 6, 8, 11, 13, 14, 22, 24, 27, 29, and 30 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Crowell in view of Huang in further view of ("The Design and Evolution of C++") by Bjarne Stroustrup (hereinafter "C++"). Applicant respectfully traverses this rejection.

Applicant's claim 16 recites

"A computer system comprising:

a system processor configured to execute instructions associated with user software;

a client coupled to said system processor via a system interconnect; and

a service processor coupled to said system processor and to said client via a maintenance bus, wherein said service processor is configured to execute application software for configuring said computer system into one or more domains and for performing diagnostics;

wherein said service processor is further configured to:

access an executable form of program instructions for manipulating said system processor and said client from one consistent state to a next consistent state;

wherein said program instructions describe a sequence of one or more transactions for manipulating said system processor and said client;

wherein said program instructions call one or more code segments that include specific information associated with said system processor and said client and wherein said program instructions are independent of said specific information; and

execute said executable form of said program instructions in response to executing said application software."

The Examiner asserts the combination of Crowell and Huang teach the features recited in Applicant's claim 16. Applicant respectfully disagrees. More particularly, Crowell discloses at paragraphs [0021 -0023]

"[0021] The present invention generally is directed to methods, articles of manufacture and systems for simulated testing of firmware without running on the actual hardware the firmware targets. In contrast to the prior art, embodiments of the present invention allow debugging of largely unmodified firmware code by providing a software simulation of targeted hardware components. The software simulation may be accomplished through the creation and use of a set of intelligent buffer objects (hereinafter "smart buffers") corresponding to registers (e.g., status, control, and results registers) in the actual targeted hardware.

[0022] When accessed, the smart buffers may call specialized functions (or behavior actions) designed to simulate behavior of the targeted hardware by modifying smart buffers associated with the same or other hardware registers, based on behavior definitions associated with the smart buffers. In other words, the behavior definitions may be developed, based on the targeted hardware specifications, to capture the cause and effect

behavior of the actual registers of the targeted hardware. As used herein, the term firmware generally describes any set of executable software code (i.e., program and data) designed to interface with (e.g., read, write, modify) hardware components.
[0023] FIG. 1 shows an exemplary computer system 100, in which embodiments of the present invention may be utilized. For example, embodiments of the present invention may be implemented as a program product for use with the system 100, to debug **firmware code 118** generally designed to configure one or more of the illustrated hardware components. The firmware code 118 may be executed by one or more system processors 112 or a service processor 113. The program(s) of the program product defines functions of the embodiments (including the methods described herein) and can be contained on a variety of signal-bearing media" (Emphasis added)

Crowell discloses at paragraph [0027]

"[0027] As illustrated, the computer system 100 generally include one or more system processors 112 and may also include a service processor 113. Each processor may be connected via a bus 114 to a memory 116, and any number of hardware components, including, but not limited to, a mass storage interface 137, I/O interface 140, and a network interface 144. The mass storage interface 137 may be operably connected to any suitable storage device, such as a direct access storage device (DASD) 138. The I/O interface 140 may include any number of hardware interface components suitable to operably connect with any type and combination of one or more I/O devices 142. The network interface 144 may be operably connected to a plurality of networked devices 146, which may include any type of networked devices, including networked-storage devices, networked printers, and other networked computers." (Emphasis added)

Crowell also discloses at paragraphs [0030-0034]

"[0030]As shown, the memory 116 may also include a simulator 130 which may be used to test the firmware code 118. For example, an objective of the firmware code 118 may be to configure various hardware components of the computer system 100, for example, during a boot process, in preparation of running the operating system 120. Further, the firmware 118 may serve **as an interface for the operating system** 120 to access hardware components, while hiding the details of such access from the operating system 120.

[0031] In any case, the simulator 130 may be generally configured to simulate the behavior of hardware targeted by the firmware 118 through the use of a set of smart buffer objects 132 that correspond to registers of the targeted hardware. The smart buffer objects 132 are generally designed

> to simulate the behavior of their corresponding hardware registers as defined by one or more behavior definitions 134. The behavior definitions 134 are generally developed in a hardware modeling process, in an effort to capture the behavior of corresponding hardware registers, for example, based on the actual specifications of the hardware components. An exemplary work flow for developing firmware 118 and hardware definitions 134 is shown in FIG. 2. ...
> [0034] For example, as illustrated in FIG. 3 conditional branches may be placed in the low level code, at step 302. Upon executing the firmware code to configure hardware, at step 304, a determination is made, at step 306, as to whether simulation is enabled (e.g., via a status flag). If simulation is not enabled, the actual hardware is accessed, at step 310. On the other hand, if simulation is enabled, rather than access the actual hardware, the firmware communicates with the simulator 130, at 308. In other words, the firmware code may access smart buffers 132 modified by the simulator 130 rather than actual hardware registers. The steps 304-310 may be repeated, for example, as long as there is more code to execute or a break point is reached, as determined, at step 312. The operations are exited at step 314. Of course, rather than place conditional branches in the actual firmware code 118, a compiler flag may be used when compiling source code, to generate firmware code that either communicates with the simulator 130 or access the actual hardware. (Emphasis added)

From the foregoing, Applicant submits Crowell is teaching using a hardware simulator routine that executes on the system to simulate the hardware using software objects called smart buffers. This enables the system firmware code (application software executing on service processor) to be run on the actual hardware without actually manipulating the hardware, but instead manipulating a software construct of the hardware (smart buffer). Thus, the smart buffers do not actually manipluate the hardware, in fact, the smart buffers merely call functions that capture the cause and effect of the actual registers in hardware. Applicant suggests the firmware code is nothing more than conventional configuration software that includes low-level code for manipulating the hardware (See above par. [0034]).

Thus, Applicant fails to see how the firmware code which has the capability of manipulating and configuring the hardware can be both the application software and the executable form of program istructions recited in Applicant's claim 16. Applicant submits it cannot, since the Applicant's application software and the program instructions

that get executed in response to the application software are two different things. Furthermore the program instructions call one or more code segments that include specific information associated with said system processor and said client **and wherein said program instructions are independent of said specific information**." This is not taught or suggested by Crowell. As noted above, the firmware of Crowell is the code that can manipulate the hardware, and the smart buffers are just simulation code objects the emulate the hardware components.

Accordingly, Applicant submits neither Crowell nor Huang teaches or suggests "wherein said service processor is further configured to: access an executable form of program instructions for manipulating said system processor and said client from one consistent state to a next consistent state," "wherein said program instructions describe a sequence of one or more transactions for manipulating said system processor and said client," "wherein said program instructions call one or more code segments that include specific information associated with said system processor and said client and **wherein said program instructions are independent of said specific information**," and "execute said executable form of said program instructions **in response to** executing said application software" as recited in Applicant's claim 16.

Accordingly, for the reasons given above, Applicant submits claim 16, along with its dependent claims patentably distinguishes over Crowell in view of Huang.

Applicant's claims 1 and 15 include features that are similar to the features recited in claim 16. Thus Applicant submits claims 1 and 15, along with their respective dependent claims patentably distinguish over Crowell and Huang for at least the reasons given above.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-41100/SJC.

Respectfully submitted,

/Stephen J. Curran/
Stephen J. Curran
Reg. No. 50,664
AGENT FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: May 14, 2007